CS 419: Production Rendering

Introduction to Monte Carlo Methods

Eric Shaffer
The Power of Randomization

- Randomization is an important tool in algorithm design.

- *Las Vegas algorithm*: uses randomness but always yields the same result for the same input.

- *Monte Carlo algorithm*: gives different results depending on "random" inputs used...gives the right answer on average.
Expected Value and Variance

- **expected value**: average value of the variable
  \[ E[x] = \sum_{j=1}^{n} x_j p_j \]

- **variance**: variation from the average
  \[ \sigma^2[x] = E[(x - E[x])^2] = E[x^2] - E[x]^2 \]

---

**throwing a die**

- **expected value**: \( E[x] = \frac{1 + 2 + \cdots + 6}{6} = 3.5 \)
- **variance**: \( \sigma^2[x] = 2.916 \)
Estimated $E[x]$

- to estimate the expected value, choose a set of random values based on the probability and average the results

$$E[x] = \frac{1}{N} \sum_{j=1}^{N} x_i$$

- bigger $N$ gives better estimates

throwing a die

- 3 rolls: $3, 1, 6 \rightarrow E[x] \approx (3 + 1 + 6)/3 = 3.33$
- 9 rolls: $3, 1, 6, 2, 5, 3, 4, 6, 2 \rightarrow E[x] \approx (3 + 1 + 6 + 2 + 5 + 3 + 4 + 6 + 2)/9 = 3.51$
Law of Large Numbers

- As the number of samples goes to infinity, the error goes to zero and the answer converges to the correct number

\[ P(E[x] = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} x_i) = 1 \]
Continuous Versions

- **expected value**

\[ E[x] = \int_a^b x \rho(x) \, dx \]

\[ E[g(x)] = \int_a^b g(x) \rho(x) \, dx \]

- **variance**

\[ \sigma^2[x] = \int_a^b (x - E[x])^2 \rho(x) \, dx \]

\[ \sigma^2[g(x)] = \int_a^b (g(x) - E[g(x)])^2 \rho(x) \, dx \]

- **estimating the expected value**

\[ E[g(x)] \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i) \]
2D Example... computing $\pi$

Use the unit square $[0, 1]^2$ with a quarter-circle

$$f(x, y) = \begin{cases} 
1 & (x, y) \in \text{circle} \\
0 & \text{else}
\end{cases}$$

$$A_{\text{quarter-circle}} = \int_0^1 \int_0^1 f(x, y) \, dx \, dy = \frac{\pi}{4}$$
2D Example...computing $\pi$

Estimate the area of the circle by randomly evaluating $f(x, y)$

$$A_{\text{quarter-circle}} \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i, y_i)$$
2D Example...computing $\pi$

By definition

$$A_{\text{quarter-circle}} = \pi/4$$

so

$$\pi \approx \frac{4}{N} \sum_{i=1}^{N} f(x_i, y_i)$$
1D Example

\[ I = \int_{a}^{b} f(x) \, dx \]

\[ \langle I \rangle = \frac{b - a}{n} \sum_{j=1}^{n} f(x_j) \quad \text{Monte Carlo estimator for integral } I \]
Central Limit Theorem

If you are sampling an average, the distribution of the average approaches the normal distribution, even if the distribution being sampled from is not normal.

- As $N$ approaches infinity, the estimate lies in a narrower band around the expected value of the integral with higher probability.
- Within three standard deviations 99.7% of the time.
- Standard deviations vary as $1/\sqrt{N}$. 
Larger N Reduces Variance
Variance Reduction

- Variance $V = \sigma^2$
- Based on what we’ve seen, if we wanted to cut error in an $N$ sample estimate in half how many samples would we need to take?
- Monte Carlo methods converge slowly
  - But often are the only realistic option
- Variance reduction techniques can help
Importance Sampling

- Do not sample uniformly
- Sample with a density that has a similar shape to the shape of \( f(x) \)
Computing the expected value of a function is an integral.

We can estimate expected value using random sampling:
- By the Law of Large Numbers
- So...we can estimate an integral by sampling.

Each integral estimate is an average.

These averages form a normal distribution:
- By the Central Limit Theorem.

The estimate will fall within 1 standard dev. with high probability:
- Can reduce the standard deviation by using more samples in the avg.
- Error reduction behaves like the function $\frac{1}{\sqrt{N}}$. 

---

Summary: Monte Carlo Integration
Stochastic Simulation with MC

- Two requirements for Monte Carlo
  - Know which probability distribution you need to sample
  - Generate sufficiently random numbers

- Random Numbers
  - Most Random Number Generators (RNGs) are pseudo-random
    - Any idea why?
  - You can get faster convergence with a quasi-random sequence
    - Doesn’t clump, samples more or less uniform across domain
      - Hammersley and Halton are examples
    - Why not just use a uniform grid of samples?